Tarea 9

Axel Báez Ochoa
Claudia Acosta Díaz

19 de octubre de 2020

1. Problema 1

Se debe probar que si r es la raíz de un árbol rojo-negro de altura h, entonces la altura negra de la raíz cumple que $h_b(r) \geq \frac{h}{2}$.

Si tomamos la raíz r del árbol rojo-negro de altura h, esta tiene color negro. Tendremos que existe al menos un camino de la raíz a un nodo NULL (llamado NIL en el código que usamos, que son los que están como "hojas"del árbol) que tiene también color negro. Supongamos que se tiene un camino como

$$r \to n_1 \to n_2 \to ... \to n_{h-1} \to NIL$$

donde $n_i, i = 1, ..., h-1$ son nodos no NULL del árbol, que llevan consecutivamente desde la raíz hasta una de las hojas NIL.

Una de las características de los árboles rojo-negros es que no puede haber dos nodos rojos consecutivos, es decir, un nodo rojo no puede tener padre ni hijo rojos. Así, sabemos que la cantidad de nodos rojos en el camino antes mencionado, que tiene h+1 nodos (contando el NIL), no puede ser mayor que $\lfloor \frac{h}{2} \rfloor$, porque además ya sabemos que los de los extremos son negros.

Por tanto, al tener que los nodos rojos no pueden superar $\lfloor \frac{h}{2} \rfloor$, se tiene directamente que los nodos negros, que determinan la altura negra, deben cumplir lo pedido, que es que son más que $\frac{h}{2}$. Así, se tiene que $h_b(r) \geq \frac{h}{2}$ como se quería probar.

2. Problema 2

Se debe probar por inducción que cualquier subarbol de un árbol rojo-negro con raíz en un nodo v, tiene al menos $2^{h_b(v)} - 1$ nodos internos.

Para el caso base, supongamos que la altura del subarbol es 0, entonces es un nodo NIL (una hoja en la forma que explican en el Cormen). Este subarbol de altura 0 con raíz $v = NIL$, tendría $2^{h_b(v)} - 1 = 2^0 - 1 = 1 - 1 = 0$ nodos internos, lo cual es consistente con la realidad pues es un árbol de un solo nodo, la raíz.

Supongamos entonces que para un subárbol de raíz x y altura negra $h_b(x)$. Tendríamos como hiótesis de inducción que $2^{h_b(x)} - 1$ es el número de nodos internos de este subarbol.

Supongamos entonces que existen dos x como los anteriores que son hijos de un nodo v. Tendremos que cada hijo de v, tendrá altura negra $h_b(x) = h_b(v)$ si el hijo es rojo o $h_b(x) = h_b(v) - 1$ si el hijo es negro. Por la hipótesis de inducción sabemos que la altura de uno de los hijos de v es $2^{h_b(x)} - 1$, pero vimos que $h_b(x)$ es al menos $h_b(v) - 1$, así, la cantidad de nodos internos de v será al menos $2^{h_b(v)-1} - 1 + 2^{h_b(v)-1} - 1 + 1 = 2^{h_b(v)} - 1$ como se quería probar.

3. Problema 3

Partiendo de los problemas 1 y 2, se debe probar que la altura de un árbol de n nodos internos cumple $h \leq 2\log_2(n+1)$.

Del problema 1 se tiene que si el árbol es de raíz r y altura h, entonces $h_b(r) \geq \frac{h}{2}$. Por tanto, usando también el problema 2, sabemos que el número de nodos internos n cumple que $n \geq 2^{\frac{h}{2}} - 1$ por lo que se probó por inducción.

Sumando 1 a ambos lados se tiene $n + 1 \geq 2^{\frac{h}{2}}$ y si se aplica logaritmo en base 2 a ambos lados, se obtiene $\log_2(n+1) \geq \frac{h}{2}$. Al multiplicar por 2 a ambos lados, se tiene lo deseado que es $2\log_2(n+1) \geq h$, o escrito en el orden contrario para que quede como se pidió, $h \leq 2\log_2(n+1)$.

4. Problema 4

Primero veamos que para arboles rojos-negros las restricciones que debe seguir son las siguientes:

1.- Cada nodo tiene un elemento (un label) llamado color, que es o rojo, o negro.

2.- La raíz del ABB es siempre negra.

3.- Los hijos de un nodo rojo tienen que ser ambos negros (o sea, no puede haber two rojos consecutivos en un camino de la raíz a una hoja).

4.- Las ligas vacías (apuntadores NULL en nodos terminales) cuentan como negro.

5.- Cualquier camino de un nodo v del árbol hacia un NULL tiene el mismo núumero de nodos negros (sin contar v). Ese numero se llama altura negra de v y lo notaremos hb(v).

Comenzamos con un árbol rojo-negro sin violaciones, y agregamos un nuevo nodo coloreado como rojo. Si ese nodo es la raíz cambiamos su color a negro por lo que la propiedad (2) no sera violada, la propiedad (1) se mantiene porque el nuevo nodo va a ser de color rojo, o negro si es la raíz. La Propiedad (5) que dice que el número de nodos negros es el mismo en cada ruta desde un nodo dado, también está satisfecho, ya que antes de insertar este nuevo nodo, el árbol cumplía esta restricción por lo que el nuevo nodo reemplaza un nodo negro NULL, y como este nuevo nodo es rojo con hijos NULL ,estos cuentan como negros según la propiedad (4) por lo que el numero de nodos negros no se vera afectado. Por lo que la única propiedad que puede fallar es la numero

(3) esto se debe a que si el nuevo nodo no es la raíz su color sera rojo, por lo que si el padre de este es rojo se violara la restricción.

5. Problema 5

Teníamos un árbol que cumplía las restricciones , hasta que se inserto un nuevo nodo que viola la propiedad (3), ya que el padre de este es rojo, si el tío de este es rojo, como el árbol cumplía las propiedades antes de la inserción del nuevo nodo, el abuelo debe de ser negro , por lo que podemos cambiar los colores del papa y el tío del nuevo nodo a negros, solucionando el problema de que el nuevo nodo y su papa eran rojos, y cambiamos el color del abuelo a rojo para mantener la propiedad (5), ahora podría ocurrir que el papa del abuelo sea rojo violando nuevamente la restricción por lo que se hará el mismo proceso pero ahora tomando al abuelo, donde el padre de este no cambiara de color, si el abuela es la raíz se violara la restricción (2), si no es la raíz se corrigió el problema y no se ah violado (2). Esto se repetirá subiendo en el arbol por lo que entonces sera $O(logn)$. Todo este proceso cambiara el color de unos nodos en cada iteración pero preserva la propiedad (5) done todo camino de un nodo hacia un NULL tienen el mismo número de negros.

6. Problema 6

Teníamos un árbol que cumplía las restricciones , hasta que se inserto un nuevo nodo que viola la propiedad (3), ya que el padre de este es rojo, si el tío de este es negro, si el nuevo nodo insertado es un hijo del lado derecho de su padre, usamos una rotación a la izquierda, para que se vuelva un hijo izquierdo, como este nuevo nodo y su papa son rojos, la rotación no afecta el numero de nodos negros para los caminos, propiedad (5), el nodo abuelo seguirá siendo el mismo , dada la forma en que esta definida la rotación, hacemos una rotación a la derecha y cambios de colores y preservamos (5) y ya que no tenemos dos nodos rojos, uno papa de otro, terminamos. Si el hijo es izquierdo este apuntara al padre que es rojo, si es derecho el padre sera negro, así que si es la raíz se solucionara el problema de el inciso 5 donde volvíamos la raíz roja. Como la altura del árbol de n nodos es $0(logn)$, le tomara a esto $0(logn)$, como ya dijimos este paso solo se repite una vez pero si sucede lo que mencionamos en el Problema 5, como mencionamos se podría repetir $O(logn)$, así que la complejidad de arreglar el árbol sigue siendo $O(logn)$, donde nunca hay mas de 2 rotaciones pues las restricciones se solucionan si entramos a lo descrito en este problema y solo continúan cuando estamos en lo descrito para el problema 5.

7. Problema 7

La solución estará en el archivo arbol2.c de la carpeta Axel-árboles dentro del mismo git de la tarea 8.